A BINARY ACCESS CONTROL SCHEME WITH SINGLE KEY

Md. Rafiqul Islam, Harihodin Selamat and Mohd. Noor Md. Sap

Faculty of Computer Science and Information System
Universiti Teknologi Malaysia, Jalan Semarak
54100 K. L, Malaysia. Tel: *6-03-2904957*, Fax: *6-03-2930933*
E-mail Address: *mmcc0004*@utmkl.utm.my.

Abstract

An access control scheme for implementing the access control matrix is presented. The proposed scheme is based upon binary form of access rights and different from the schemes that are based on the concept of key-lock pairs. In this scheme each user is assigned one key. The keys are possessed by the user, and can be used to derive the access rights to the files. The scheme can easily handle the dynamic access control problem, such as changing access right, adding a user or file and deleting a user or file. This scheme is more effective and efficient for the systems where files are accessible to only limited number of users.

Keywords: Access right, dynamic access and single key.

1. Introduction

Data protection is very important issue in a computer system, because of the increasing complexity of various sorts of information, the large number of users, and the widely used communication networks. The access control system can be used to prevent the information stored in a computer from being destroyed, altered, disclosed or copied by unauthorized users. The access matrix is a conceptual model [2, 10] that specifies the rights that each user possesses for each file. There is a row in this matrix for each user, and a column for each file. Each cell of the matrix specifies the access authorized for the user in the row to the file in the column. The task of access control is to ensure that only those operations authorized by the access matrix actually get executed. An example of an access matrix is shown in Fig. 1. We assume that all access rights are expressed by numerals. Linear hierarchy of access privileges may be applied here. That means, the right to read implies the right to execute, the right to write implies the

rights to read and execute and so on. In the access matrix shown below, the user U_1 can execute the file F_1 and read the file F_2 and U_3 can delete the file F_2.

Files / Users	F_1	F_2	F_3	F_4
U_1	1	2	0	4
U_2	2	0	3	0
U_3	0	4	0	2

0: No access
1: Execute
2: Read
3: Write
4: Delete

Fig. 1 An access control matrix .

Wu and Hwang [3] proposed an alternative scheme storing just one key for each user and one lock for each file. In their system lock is a function of key and the access right of the user is computed by operations on lock and key. Several relevant systems [4-8] appeared in the literature after Wu and Hwang's work. Chang *et al.* in [9] have improved the results of Hwang *et al.*'s scheme. These proposed methods theoretically work well. However, on the dynamic access control such as adding or deleting files or users, the keys or locks should be reestablished. Besides the complexity of operations on generating keys and locks is sophisticated and time consuming. Jan in 1987 proposed a single key access control scheme [11] and Tseng *et al.* in 1990 proposed another single key access control scheme with high data security [12]. These two schemes are simple, but they need complex computation for generating keys as well as to find the access rights. In case of addition or deletion of a file the schemes require to recompute almost all the keys. In this paper we proposed a single key access control scheme that is based on binary form of access rights. The proposed scheme is very simple and in case of dynamism of addition or deletion of a file it is more efficient than that of other single key methods. The key construction and access right calculation processes are simple. Furthermore the scheme gives flexibility to use of access modes that in other single key systems creates problem. In next section we briefly review the single key methods.

2. The Single Key Methods

The key construction process of Jan's scheme [11] is as follows:

$$K_i = \sum_{j=1}^{n} r_{ij} \cdot T^{j-1} \qquad (2.1)$$

where, K_i is the key of the user U_i, r_{ij} denotes the access right of the user U_i to the file F_j, n is the total number of files in the system, T is the total number of access rights (including no access). The access right of the user is computed as follows

$$r_{ij} = \left\lfloor \frac{K_i}{T^{j-1}} \right\rfloor mod\ T \qquad (2.2)$$

Here we can see that the key construction process depends on some terms that are powers of T. If T becomes reasonably big, the key construction process will be complex. On the other hand, if the total number of access rights is changed, the system must be reconstructed. In case of file deletion or addition almost all the keys should be reconstructed.

Tseng *et al.* proposed an access control scheme with high data security that is based on the generalized Markel-Helman cryptosystem [12]. They paid attention to protecting the access control information itself. In this system the key construction process is as follows:

$$K_i = \sum_{j=1}^{n} r_{ij} A_j \qquad (2.3)$$

where $A'_j = A_j.w \bmod d$, $A_j = T^j - 1$, $d > T^n - 1$, $w < d$ and gcd $(w, d) = 1$.

In Tseng *et al.*'s scheme the key construction process is more complicated than that of Jan's method. The access right is computed as follows

$$r_{ij} = \left\lfloor \frac{K'_i}{T^{j-1}} \right\rfloor mod\ T \qquad (2.4)$$

where $K'_i = K_i.\theta \bmod d$ and θ is an integer such that

$$w.\ \theta \bmod d = 1 \qquad (2.5)$$

The solution of the equation (5) is as below:

$\theta = Inv\ [w, d]$, Inv denotes inverse of (w, d) that can be found by extended Euclid's algorithm [1]. To find any access right we have to compute θ and hence computation of the access right is not efficient that is main thing in an access control system. In case of file addition or deletion this system also needs to recompute almost all the keys. In the next section we propose a simple and efficient single key access control scheme that gives better results than that of the previously proposed single key methods.

3. The Binary Single Key Method

We will describe the proposed method with respect to key construction process, checking access right and dynamic access control, such as changing access right, adding/ deleting a user or file. In first subsection we describe the basic concept that includes the key initialization process and verification of access right. In the second subsection we describe the dynamic access control.

3.1 Basic Concept

Let each access right a_{ij} in access matrix is represented in its binary form $a_{ij} = (a_{ij}^c a_{ij}^{c-1} \ldots a_{ij}^1)$ where, $c = 1 + \lfloor a_{max} \rfloor$ and a_{max} is the maximum of access rights. Suppose there are m users and n files in the system. In the proposed method each user is assigned a key that is computed from the binary form of access rights. The key is possessed by the user and can be used to derive the access rights to the files. Suppose K_i denotes the key of the user U_i and the key is represented as $K_i = (K_i^c, K_i^{c-1}, \ldots, K_i^1)$, *i.e.*, each key is broken into c elements. Now we compute each element of the key K_i as follows:

$$K_i^r = \sum_{j=1}^{n} a_{ij}^r \cdot 2^j \quad \text{for } r = 1, 2, \ldots, c \text{ and } i = 1, 2, \ldots, m. \qquad (3.1)$$

where a_{ij}^r denotes the rth bit of a_{ij} and $a_{ij}^r \in \{0, 1\}$.

Suppose $c = 3$, then we can compute the elements of the key K_i (the key of the user U_i) as below:

$$K_i^1 = \sum_{j=1}^{n} a_{ij}^1 \cdot 2^j$$

$$K_i^2 = \sum_{j=1}^{n} a_{ij}^2 \cdot 2^j \qquad (3.2)$$

$$K_i^3 = \sum_{j=1}^{n} a_{ij}^3 \cdot 2^j$$

for $i = 1, 2, \ldots, m$ and where a_{ij}^1 denotes first bit of the access right a_{ij}.

From the above system of equations in (3.2) we found that we have to compute c elements for each key vector.

Example 1: Key construction process

By considering the access control matrix in Fig. 1 and using binary form of access rights, it can be found a binary access control matrix shown in Fig. 2.

Files / Users	F_1	F_2	F_3	F_4
U_1	001	010	000	100
U_2	010	000	011	000
U_3	000	100	000	010

Fig. 2. The binary access control matrix for Fig. 1.

By considering access control matrix in Fig. 2, using (3.1) the key of the users are as follows:

For user U_1:

$$K_1^1 = 2^1 = 2,\ K_1^2 = 2^2 = 4,\ K_1^3 = 2^4 = 16,$$

$$K_1 = (K_1^3, K_1^2, K_1^1) = (16, 4, 2).$$

For user U_2:

$$K_2^1 = 2^3 = 8,\ K_2^2 = 2^1 + 2^2 = 9,\ K_2^3 = 0,$$

$$K_2 = (K_2^3, K_2^2, K_2^1) = (0\ 9, 8).$$

For user U_3:

$$K_3^1 = 0,\ K_3^2 = 2^4 = 16,\ K_3^3 = 2^2 = 4,$$

$$K_3 = (K_3^3, K_3^2, K_3^1) = (4, 16, 0).$$

Therefore, $K_1 = (16, 4, 2)$;

$K_2 = (0, 9, 8)$;

$K_3 = (4, 16, 0)$.

We can compute the access right of the user U_i to the F_j by using the following formula:

Compute
$$a_{ij}^r = \left\lfloor \frac{K_i^r}{2^j} \right\rfloor \bmod 2 \quad \text{for } r = 1, 2, \ldots, c \qquad (3.3)$$

Example 2: Access right verification

Let the user U_2 sends a delete (in numeric 3) request to the file F_3. Now we can verify whether the user U_2 is allowed to delete the file F_3 or not. Here we have to find a_{23} = ? Using formula (3.2) we find

$$a_{23}^1 = \left\lfloor \frac{8}{2^3} \right\rfloor \bmod 2 = 1,$$

$$a_{23}^2 = \left\lfloor \frac{9}{2^3} \right\rfloor \bmod 2 = 1,$$

$$a_{23}^3 = 0,$$

That is, $a_{23}=(a_{21}^3 a_{23}^2 a_{23}^1)=(011)=3$, which is correct. Hence the request of the user U_2 is allowed.

If the user U_3 wants a read access to the file F_1, we have to find a_{31} = ? From formula (3.3) we get

$$a_{31}^1=\left\lfloor \frac{4}{2^1} \right\rfloor mod\ 2=0,$$

$$a_{31}^2=\left\lfloor \frac{16}{2^1} \right\rfloor mod\ 2=0,$$

$$a_{31}^3=0,$$

So, the request is denied, since $a_{31} = 0 \neq 1$.

3.2. Dynamic Access Control

In this subsection we describe the dynamic access control process that means changing access right, adding/deleting a user or file.

1) Changing access right

Let us consider access right a_{ij} is changed to $b_{ij}=(b_{ij}^c b_{ij}^{c-1} \ldots b_{ij}^1)$. By executing ***algorithm 1***, we can update the key.

Algorithm 1: ***Changing access right***

Input: K_i, a_{ij} and b_{ij};

Output: K'_i.

Step 1: find access right a_{ij} using formula (3.3);

Step 2: Input access right b_{ij};

Step 3: **For** $1 \leq r \leq c$ **do**

```
begin
  set t1 = b_ij^r and t2 = a_ij^r ;
  compute t = t1 - t2;
        If (t ≠ 0) then
           K_i^r = K_i^r + t.2^j ;
        Else
           K_i^r = K_i^r ;
  end_for;
```

Step 4: K'_i.

Example 3: Changing access right

Suppose the access right $a_{12} = 010$ will be changed to $b_{12} = 011$. Here $K_1 = (16, 4, 2)$. By executing the *algorithm 1*, we get $K'^1_1 = K^1_1 = 16$, $K'^2_1 = K^2_1 = 4$, $K'^3_1 = K^3_1 + 2^2 = 2 + 4 = 6$, $K'_1 = (16, 4, 6)$.

Let $a_{34} = 010$ will be changed to $b_{34} = 100$. Here $K_3 = (4, 16, 0)$. So, by executing *algorithm 1*, we get $K'^1_3 = K^1_3 + 2^4 = 4 + 16 = 20$, $K'^2_3 = K^2_3 - 2^4 = 16 - 16 = 0$, $K'^3_3 = K^3_3 = 0$, $K'_3 = (20, 0, 0)$.

If we wish to verify any access right with changing values of the keys, the result will be correct.

2) Adding and deleting file

Let the file F_q will be added to the system. By executing the following *algorithm 2*, we can update the keys of the users.

Algorithm 2: Adding file

Input: F_q, access rights of the users to the file F_q.

Output: Updated keys of the users.

```
Begin
 /* begin of the algorithm */
   For 1 ≤ i ≤ m do
      begin
          For 1 ≤ r ≤ c do
             begin
               t = a^r_iq;
                If ( t ≠ 0) then
                        K'^r_iP = K^r_iP + t.2^j;
               Else
                        K'^r_iP = K^r_iP ;
             end_for;
      end_for;
  end_of_algoritm.
```

In case of deletion of a file the system uses similar algorithm by changing the statement $K'^r_{iP} = K^r_{iP} + t.2^j$ as $K'^r_{iP} = K^r_{iP} - t.2^j$.

3) Adding and deleting users

In this system the process of addition or deletion of a user is very easy. When a user is added, the system will constrict the key for the new user. If a user is deleted from the system, we can delete the key for the user.

5. Discussions

Here we discuss the practical issue and the efficiency of the proposed scheme briefly.

5.1. Practical Issue

As we know each key of a user consists of c elements. If we take one integer for one element, the key can be defined as a structure (record) of c integers. However, one integer may not be enough for storing one element of the key. For instance we consider *32* bit computer, the largest integer allowed by such a computer is 2^{32}. Since each element is a sum of some terms that are in as power of 2, it may occur overflow to hold one element using one integer. In such a case we must take several integers for each element. So, we require array of integers for holding one element. Thus each element of a key is an array of several integers and each key is a structure of such c arrays.

5.2. The Efficiency of the Scheme

First, the construction of the key element of the proposed binary access control scheme is simple, since the key is a sum of some terms that are in the form of power of *2*. Here we need to consider only the non-zero bits of the access right (*i.e.*, for $a_{ij}^r=1$). As we know the access matrix is usually a sparse [2, 9, 10] and we do not need to consider the zero access rights as well as the zero bits of non-zero access rights, the key construction process is easy. Second, to find out each bit of the access right the system requires *2* divisions. So, to find c bits of a key it requires $2c$ divisions and c *is* usually a very small number, such as $c = 3$ or $c = 4$. Third, dynamic access control, that is, changing access right, adding or deleting a user or file can be easily achieved. In case of addition or deletion of a file the system requires to update the elements of the key only for $a_{ij}^r=1$ of those users who can access the file to be added or deleted. Fourth, the storage required to implement the proposed scheme is $O(cm) = O(m)$, that is, one key for one user. Fifth, If we consider $c = 4$ and construct the system, we can accommodate $2^4 - 1 = 15$ access modes (execute, read and so on). That would be enough for practical use. In Jan's as well as Tseng *et al.*'s schemes considering $T = 15$ increase the overhead of computations. Furthermore, in Tseng *et al.*'s scheme if T is taken as small as possible, after considering one

more access mode the system should be reconstructed. However the proposed scheme gives flexibility to use of access modes.

7. Conclusion

In this paper we proposed a very simple and efficient scheme based on binary access modes. For the proposed method we devise algorithms for implementation of dynamic access control, such as changing access right and updating file. One good feature of our system is that insertion or deletion of any file can be successfully implemented without reconstructing all the key-vectors. The storage required to implement our scheme is small. The proposed method gives the flexibility to use of access modes. Furthermore, the proposed method is very suitable for implementing sparse access control matrix.

References

1. D. E. R. Denning, *Cryptography and Data Security*; Addison-Wesley, Reading, MA, 1983.
2. G. S. Graham and P. J. Denning, *Protection- Principle and Practice*; Proc. Spring Joint Computer Conf., Vol. 40, AFIPS Press, Montvale, NJ, 1972, pp. 417-429.
3. M. L. Wu and T. Y. Hwang, *Access Control with Single-Key-Lock*; IEEE Transaction on Software Engg., Vol. SE-10, No. 2, 1984, pp. 185-191.
4. C. C. Chang, *On the design of a key-lock-pair mechanism in information protection systems*; BIT, Vol. 26, 1986, pp. 410-417.
5. C. C. Chang, *An Information Protection Scheme Based upon Number Theory*; The Computer Journal, Vol. 30, No. 3, 1987, pp. 249-253.
6. C. K. Chang and T. M. Jiang, *A Binary Single-Key-Lock System for Access Control*; IEEE Transaction on Computers, Vol. 38, No. 10, 1989, pp. 1462-1466.
7. J. J. Hwang, B. M. Shao and P.C. Wang, *A New Access Control Method Using Prime Factorization*; The Computer Journal, Vol. 35, No. 1, 1992, pp. 16-20.
8. C. C. Chang, J. J. Shen and T. C. Wu, *Access control with binary keys*; Computers & Security, Vol. 13, 1994, pp. 681-686.
9. C. C. Chang, D. C. Lou and T. C. Wu, "A Binary Access Control Method using Prime Factorization", Information Sciences, 1997, pp. 15-26.
10. R. S. Sandhu and P. Samarati, "Access control: principle and practice", IEEE communication magazine, 1994, pp. 40-48.

11. J. K. Jan, 'A single key access control scheme in information protection system, Proceedings of National computer Symposium, Taiwan, 1987, pp. 299-303.

12. J. C. R. Tseng and W.-P. Yang, *A new access control scheme with high data security*, ninth annual international phoenix conference on computer and communications, IEEE comp. soc. press, 1990, pp. 683-688.

13. M. R. Islam, H. Selamat and M. N. M. Sap, "A dynamic access control with key-pair", *Malaysian Journal of Computer Science*, Vol. 10, No. 1, 1997, pp. 36-41.